2Б

УДК 681.3

М.К. Буза
Белорусский государственный университет, г. Минск, Беларусь
Беларусь, г. Минск, пр-т Независимости 4, bouza@bsu.by

Эффективный протокол для распределенных систем

M.K. Bouza
Belarusian State University
Nezavisimosti av., 4, Minsk, Belarus, bouza@bsu.by

Efficient Protocol for Distributed Systems

М.К. Буза
Білоруський державний університет, м. Мінськ, Білорусь
Білорусь, м. Мінськ, пр-т Незалежності 4, bouza@bsu.by

Ефективний протокол для розподілених систем

Разработан широковещательный протокол быстрой и надежной передачи данных. Логически он позволяет работать с кластером узлов как с одним объектом. Исследованы возможности его применения для повышения эффективности систем распределенной обработки данных.
Ключевые слова: компьютерная сеть, широковещательные протоколы, распределенная обработка, кэширование, целостность данных

Developed a broadcast protocol for fast, reliable data transfer. Logically it allows you to work with the cluster node as a single object. Explores the possibility of its application for increase of efficiency of systems of distributed data processing.
Key words: computer network, broadcast protocols, distributed processing, caching, data integrity.

Розроблено широкомовний протокол швидкої і надійної передачі даних. Логічно він дозволяє працювати з кластером вузлів як з одним об’єктом. Досліджено можливості його застосування для підвищення ефективності систем розподіленої обробки даних.
Ключові слова: комп’ютерна мережа, широкомовні протоколи, розподілена обробка, кешування, цілісність даних.

Введение

На сегодняшний день компьютерные сети получили многофункциональное использование. Одним из основных требований, предъявляемых к их работе, является быстрая и надежная передача информации. Особую важность приобретает это требование, если речь идет о широковещательном режиме работы.

В современных компьютерных сетях существуют два типа технологий передачи:

– широковещательные сети;

– сети с передачей от узла к узлу.

Широковещательные сети обладают единым каналом связи, совместно используемым всеми узлами сети.

Эти сети позволяют передавать данные одновременно всем узлам. Такая операция называется широковещанием (broadcast).

Сети с передачей от узла к узлу состоят из большого количества пар соединенных узлов. В таких сетях передача данных реализуется через промежуточные узлы. Операции в сетях и соответствующие им протоколы называются уникальными (unicast), хотя может существовать несколько возможных путей от источника до получателя [1].

Широковещательные протоколы, реализующие однократное дублирование информации в сети с возможностью множественного доступа к этой информации многими узлами сети, для пользователей имеют ряд привлекательных черт:

– устраняется множественное дублирование информации, которое снижает скорость работы как всей сети, так и приложений, которые выполняются на многих узлах сети;

– легче контролировать один направленный поток, чем множество потоков;

– в случае групповой рассылки требуются дополнительные протоколы.

Все это наводит на мысль использовать эти протоколы в системах распределенной обработки данных для повышения их эффективности [2], [3].

В распределенной обработке данных (РОД) в последнее время появляются новые концепции обработки и новые технологии. Среди них можно обозначить Grid-системы и «облачные» технологии.

Распределенные вычисления незаменимы, например, при переборных задачах: поиск простых чисел, молекул и их сочетаний при создании лекарственных веществ, обработка данных во многих сферах деятельности [4], [5]. Специфической особенностью РОД является налаживание интерфейса между различными обработчиками данных, географически удаленными друг от друга, и интегрирование результатов обработки.

Две основные причины порождают необходимость использования РОД. Во-первых, природа приложения может требовать применения коммуникационной среды. Во-вторых, необходимость применения РОД может диктоваться экономическими соображениями. Кроме того, распределенные системы значительно проще масштабировать.

В связи с этим была поставлена **цель работы –** создать широковещательный протокол, который бы быстро и надежно транспортировал данные в системах распределенной обработки.

Анализ протоколов передачи данных

Был проанализирован ряд протоколов для изучения их возможностей и технологий проектирования. Выявлены их достоинства и недостатки. Среди таких протоколов, в частности, были рассмотрены:

– IGMP (Internet Group Management Protocol), используемый при проведении видеоконференций, передаче звуковых сообщений, а также группового исполнения команд различными компьютерами;

– RTP (Transport Protocol for Real – Time application), предназначенный для доставки данных в реальном масштабе времени. Приложение обычно используют RTP поверх протокола UDP, чтобы использовать его возможности мультиплексирования и контрольного суммирования. В принципе RTP может использоваться и поверх любой другой транспортной среды [6];

– RTCP (RTP Control Protocol), базирующийся на периодической передаче управляющих пакетов всем участникам сессии, используя тот же механизм рассылки, что и для пакетов данных. Главная задача протокола – обеспечение обратной связи для контроля качества при рассылке данных;

– RTSP (Real – Time Streaminy Protocol) – двунаправленный протокол, обеспечивающий согласование эффективной доставки по IP-сетям для приложений типа «один – многим». Архитектурно RTSP расположен над RTP и RTCP;

– RSVP (Resource Reservation Setup Protocol) используется, чтобы запросить для приложения определенный уровень качества сетевых услуг QoS вдоль всего

маршрута транспортировки данных, обеспечивая резервирование необходимых сетевых ресурсов. Механизм обеспечения QoS включает в себя классификацию пакетов, административный контроль и диспетчеризацию;

– UDP (User Datagram Protocol) проектировался для создания в объединенной системе компьютерных сетей с коммутацией пакетов режима передачи дейтаграмм клиента. UDP предполагает, что нижестоящим протоколом является IP. UDP обеспечивает широковещательную рассылку данных , но не надежен. Поэтому приложения, требующие гарантированного получения потоков данных, должны использовать протокол TCP.

Были исследованы и ряд других протоколов и на основе их разработан протокол UBTP (Universal Broadcast Transport Protocol), обеспечивающий требуемое качество передачи информации.

Концепция построения протокола UBTP

Предлагается создать на базе протоколов TCP\IP универсальный протокол для работы с группой хостов, как с логически одним объектом. Идея его создания основана на использовании обоих протоколов стека TCP\IP UDP и TCP. Передача наиболее объемной информации ведется по каналу UDP, а обработка ошибок передачи и выполнение служебных задач реализуется на базе протокола TCP. Будем называть этот протокол UBTP (Universal Broadcast Transport Protocol).

В общем случае протокол UBTP решает три основные задачи:

– передача пакетов по протоколу UDP в широковещательном (Broadcast) режиме;

– кэширование информации;

– управление передачей пакетов.

Реализация протокола UBTP включает следующие компоненты:

Broadcast Server – выполняет подключение и отключение клиентов от канала передачи данных в Broadcast режиме. Передача Broadcast потока, создание обработчика ошибок, восстановление последовательности данных после сбоя в Broadcast потоке;

Broadcast Client – выполняет подключение к Broadcast Server и получение данных из Broadcast потока UDP или от обработчика ошибок сервера.

Возможны две схемы взаимодействия клиента и сервера:

– сервер и клиент обмениваются уведомлениями о начале и конце сессии;

– клиент получает доступ к потоку данных и не уведомляет сервер о событиях, связанных с данными.

Первая схема предпочтительна для тех приложений, где требуется достичь качества передачи с коэффициентом потерь, равным нулю. Для мультимедийной информации предпочтительнее вторая схема взаимодействия [7].

Прием-передача ведется в промежуточные буфера непрерывно. Отметим, что протоколы реагируют на качество сетевых интерфейсов. Чем качественнее сеть, тем меньше памяти и ресурсов необходимо протоколу. По заполнению одного из буферов передача будет заблокирована до тех пор, пока приложение не заберет данные из буфера, поэтому следует правильно определять размеры приемных буферов, исходя из качества сети.

Восстановление целостности данных после сбоя реализуется двумя принципиально разными схемами, которые могут использоваться совместно. В этих схемах имеется одно общее – передатчик не прекращает посылку UDP пакетов до передачи всей подготовленной информации. Приемник же может вести себя по-разному.

Если приемник обнаружил потерю целого диапазона пакетов, он передает обработчику ошибок передатчика номера пакетов этого диапазона, приостанавливая при этом передачу до получения потерянных пакетов по TCP, после чего опять возобновляется прием по UDP. Приемник, обнаружив потерю целого диапазона пакетов, передает передатчику параметры этого диапазона, не прекращая прием. Приемник ведет таблицу потерянных пакетов из диапазона, которые восстанавливаются по мере параллельной передачи информации по UDP и TCP. Передатчик также ведет таблицу запросов на восстановление данных, и по мере нахождения потерянных данных в кэше передаёт их приемнику по TCP, не прекращая вещание по UDP. Передатчик ведет таблицу качества связи с каждым хостом, участвующим в сеансе и может, в случае постоянных проблем, перевести хост на передачу по протоколу TCP поток данных, дублируя их.

Для синхронной работы многих хостов необходимо использовать первую схему. Вторая схема предпочтительна для систем распределенных вычислений, так как синхронизация нужна только при инициализации распределенных вычислений, а информация должна распространяться в сети с максимальной скоростью. Общий принцип взаимодействия по протоколу UBTP представлен на рис. 1. Схема реализации протокола приведена на рис. 2.

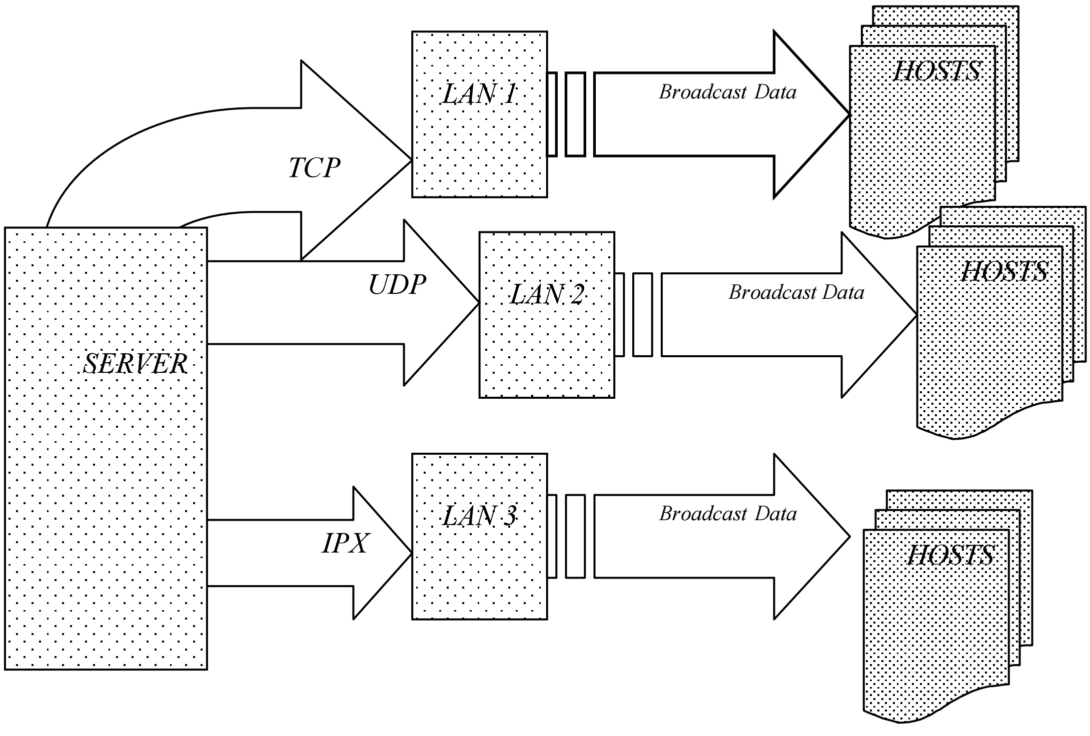


Рисунок 1 – Общий принцип трансляции потока данных по сегментам сети

Структура Broadcast Server

Broadcast Server состоит из трех независимых блоков, работающих параллельно:

– блок передачи информации по UDP;

– блок кэширования информации;

– блок восстановления информации.

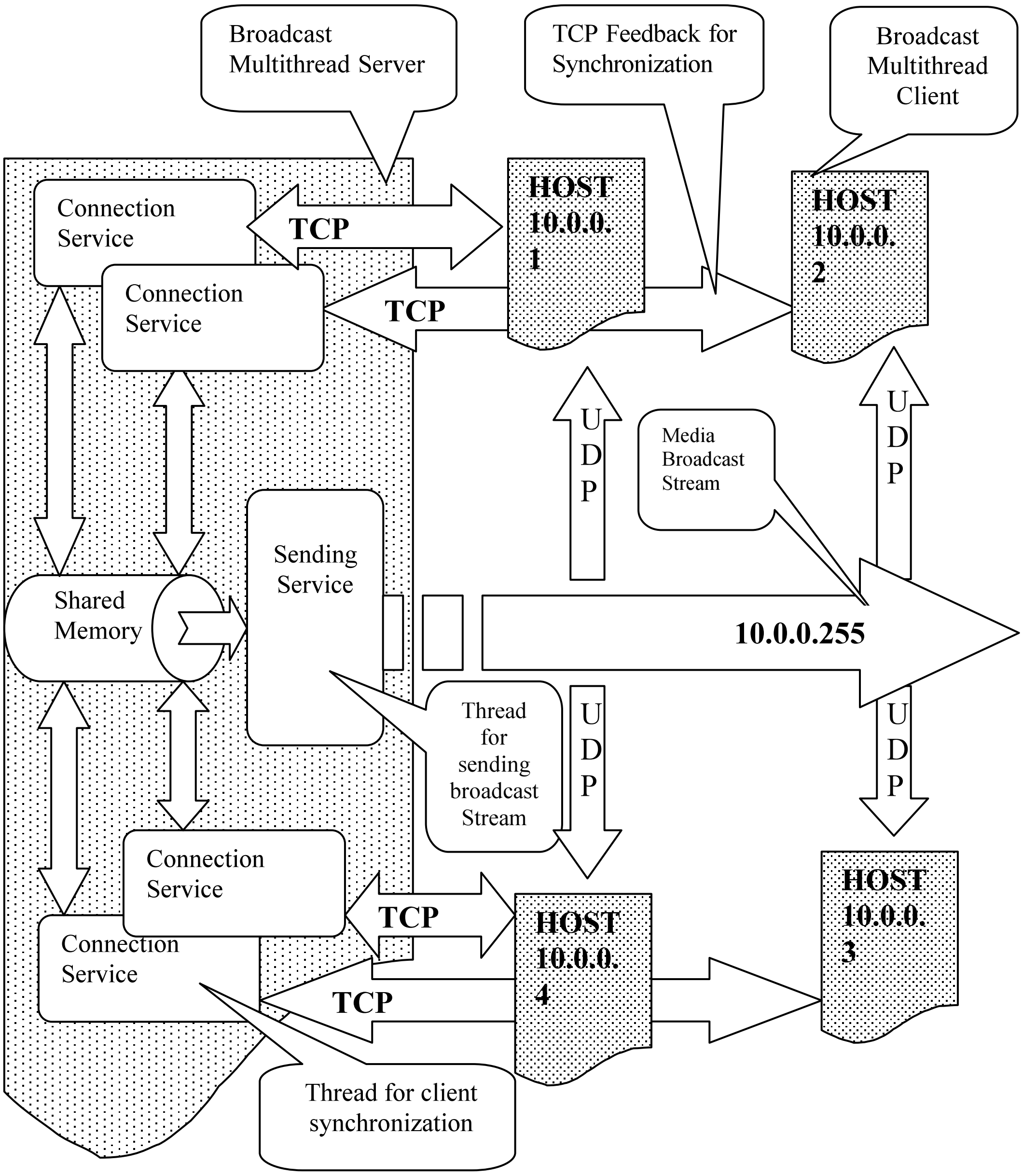


Рисунок 2 – Схема реализации протокола UBTP

Восстановление информации по TCP ведется в 2 этапа:

– поиск потерянного интервала информации в кэше;

– передачи этого интервала клиенту по TCP.

Восстановление информации невозможно без наличия её копии, организованной специальным образом. Эту копию в дальнейшем мы будем называть кэшем или таблицей истории информации.

Кэш и таблица имеет кольцевую структуру. При достижении ее конца, таблица начинает перезаписывать себя с начала. Таким образом, имеется кольцо, в котором дублируется передаваемая информация за предопределённый интервал времени. Любую информацию за этот интервал времени, таким образом, можно восстановить.

Интервал времени настраивается в процессе работы передатчика и зависит от качества сети.

Если потери начинают возрастать, то интервал времени увеличивается, и наоборот. Информация в кэше проиндексирована по времени передачи и хранится в заголовке кэша.

От алгоритма поиска информации в кэше зависит эффективность протокола. Поиск ведётся с использованием индексов, созданных на этапе построения кэша. Имея в запросе на восстановление информацию о самом раннем и о самом позднем пакете потерянного диапазона, методом дихотомии приближаемся к требуемому диапазону пакетов и отправляем их клиенту, иначе посылается признак ошибки (NO_DATA_IN_HASH). В этом случае клиент сам решает вопрос о продолжении участия в группе хостов.

Функции Broadcast Client

Broadcast Client выполняет 3 основные задачи:

– прием информации по UDP в системный буфер;

– восстановление информации в случае сбоя;

– проверку информации на непрерывность.

Прием информации выполняется в системный буфер, где она записывается последовательно и непрерывно независимо от потери какого-либо интервала данных. Если при приеме обнаруживается разрыв, записывается точка разрыва данных. Например, если в системном буфере может быть записано (0x100[0x100], 0x0a0c100[0xb00]...), то при получении данных имеются 2 дыры по адресам 0x100, 0x0a0c100 с размерами [0x100] и [0xb00]. Прием информации по UDP в системный буфер ведется непрерывно.

Восстановление данных при сбое происходит параллельно с приемом первым сервисом информации и заполнении системного буфера информацией о дырах.

Как только появилась дыра в данных, сервис отправляет на сервер запрос с указанием интервала и ждет ответ.

В случае наличия данных на сервере, он принимает данные (по TCP) и закрывает дыру, иначе помещает в системный буфер ошибку (MISSING_DATA). И так далее до закрытия всех дыр.

После получения последней порции информации от сервера происходит проверка ее на целостность – просто проверяется отсутствие ошибки MISSING_DATA в буфере.

При положительном исходе данные становятся доступными для дальнейшей обработки клиентом.

Программная реализация протокола представляет собой две библиотеки – клиентскую и серверную.

Они могут включаться в приложения как DLL или LIB компоненты.

Проведенное тестирование скорости и качества передачи информации в сети показали, что в среднем из 200 Мбайт данных, переданных по UDP, на долю протокола TCP приходится в среднем до 10 Мбайт при загрузке сети 50%. Результаты тестирования, полученные на сегменте LAN 10Base–T, приведены в табл. 1.

Графа «Потери» не означает потерю информации.

В соответствии с концепцией, это та доля информации, для передачи которой привлекается протокол TCP. Большой процент передач по TCP объясняется низкой пропускной способностью сегмента Ethernet.

Следует подчеркнуть, что при использовании разработанного протокола скорость работы со многими клиентами приблизилась к скорости работы системы с одним клиентом.

Таблица 1 – Результаты тестирования протокола

Объем, (Мбайт)	Загрузка сети, (%)	Потери, (Мбайт)	Количество клиентов
200	10	2	10

200	50	7	10
200	80	13	10

Таблица 2 – Передача мультимедийных кадров

Объем кадра (Мбайт)	Загрузка сети (%)	Протокол на основе UDP		TCP	Количество клиентов
		Потери (Мбайт)	Количество кадров в секунду у каждого клиента	Количество кадров в секунду у каждого клиента	
1.5	10	0.1	100	80	1
1.5	10	0.156	80	10	10
1.5	50	0.2	60	10	10
1.5	80	0.4	55	3	30

Следующий тест, эмулирующий передачу мультимедийной информации кадрами фиксированного размера (1,5 Мбайт), демонстрирует преимущества данной концепции.

Результаты теста приведены в табл. 2.

В тесте сравнивается количество полученных кадров каждым пользователем в секунду при использовании данной концепции на базе основного протокола UDP с восстановлением потерянной информации с помощью протокола TCP и передачей кадров с использованием только протокола TCP.

Выводы

В работе рассмотрены основные требования к распределенной обработке. Проанализированы многие протоколы, используемые в компьютерных сетях. На основе семейства протоколов TCP/IP создан комбинированный широковещательный протокол UBTP, позволяющий быстро и надежно передавать информацию. Апробирование разработанного протокола осуществлено в ряде экспериментов, которые подтвердили его достоинства.

Литература

1. Таненбаум Э. Компьютерные сети / Таненбаум Э. – Спб. : Питер, 2002. – 848 с.
2. Буза М.К. Методы и средства распределенной обработки данных / М.К. Буза, Л.Ф. Зимянин // Выбраныя навуковыя працы Беларускага Дзяржаунага Універсітэту : у 7 т. – Минск : БДУ, 2001. – Т. 6 : Матэматыка. – С. 92-116.
3. Цимбал А.А. Технология создания распределенных систем /А.А. Цимбал, М.Л. Аншина. – Спб. : Питер, 2003. – 576 с.
4. Буза М.К. Проектирование распределенных вычислений на основе DSM – моделей / М.К. Буза, Л.Ф. Зимянин // Информационные системы и технологии (IST'2002) : материалы первой междунар.конф. : в 2 ч. – Минск : БГУ, 2002. – Ч. 2. – С. 118-122.
5. Eastlake D. Randomness Recommendations for Security / D. Eastlake, S. Crocker, J. Schiller. – RFC 1750, DEC, Cybercash, MIT. – December 1994.
6. Mark Baugher. Real-Time Transport Protocol Management Information Base / Mark Baugher, John Du, Stan Naudus [Электронный ресупс]. – Режим доступа : ftp://ftp.ietf.org/internet-drafts/draft-ietf-avt-rtp-mib-00.txt.

7. Floyd S. The synchronization of periodic routing messages / S. Floyd, V. Jacobson // SIGCOMM Symposium on Communications Architectures and Protocols (San Francisco, California). – ACM, Sept. 1993. - P. 33-44.

Literatura

1. Tanenbaum E.Komputernie seti. – Spb.: Piter, 2002. 848 s.
2. Bouza M.K. Metodi i sredstva raspredelennoi obrabotki dannih /M.K. Bouza, L.F. Zimyanin //Vibraniya navukoviya praci Belaruscaga Dyarjaunaga Universitetu: u 7 t. T. 6 Matematika – Minsk: BDU. 2001. – s. 92-116
3. Cimbal A.A. Tehnologiya sozdaniya raspredelennih sistem. /A.A. Cimbal, M.L.Anshina. – Spb.: Piter, 2003. – 576 s.
4. Bouza M.K. Proectirovanie raspredelennih vichislenii na osnove DSM – modelei M.K. Bouza, L.F. Zimyanin //Informacionnie sistemi i tehnologii (IST'2002): Materiali pervoi mejdunar.konf.: v 2ch. Ch.2 – Minsk.: BGU, 2002 – s. 118-122.
5. Eastlake D., Crocker S., Schiller J. Randomness Recommendations for Security, RFC 1750, DEC, Cybercash, MIT, December 1994.
6. Mark Baugher, John Du, Stan Naudus. Real-Time Transport Protocol Management Information Base, ftp://ftp.ietf.org/internet-drafts/draft-ietf-avt-rtp-mib-00.txt.
7. Floyd S., Jacobson V. The synchronization of periodic routing messages //SIGCOMM Symposium on Communications Architectures and Protocols (San Francisco, California), pp. 33-44, ACM, Sept. 1993.

RESUME

M.K. Bouza

Efficient Protocol for Distributed Systems

Developed a broadcast protocol for fast, reliable data transfer. It makes logical point of view to work with the cluster node as a single object. We investigate the possibility of its application for increase an efficiency of distributed data processing systems. The software support of protocol was implemented as a library functions written in Visual C++.

Статья поступила в редакцию 19.04.2013.